Statistical Model Checking: Past, Present, and Future

Kim Guldstrand Larsen, Axel Legay

Statistical Model Checking: Past, Present, and Future

Kim G. Larsen and Axel Legay

Aalborg University
Inria

Abstract. Statistical Model Checking (SMC) is a compromise between verification and testing where executions of the systems are monitored until an algorithm from statistics can produce an estimate for the system to satisfy a given property.
The objective of this introduction is to summarizes SMC as well as a series of challenges for which contributors at Isola propose a solution. Contributions include new SMC toolsets, new flexible SMC algorithms for larger classes of systems, and new applications.

1 Introduction

Computers play a central role in modern life and their errors can have dramatic consequences. For example, such mistakes could jeopardize the banking system of a country or, more dramatically, endanger human life through the failure of some safety systems. It is therefore not surprising that proving the correctness of computer systems is a highly relevant problem.

The most common method to ensure the correctness of a system is *testing* (see [BJK+05] for a survey). After the computer system is constructed, it is tested using a number of *test cases* with predicted outcomes. Testing techniques have shown effectiveness in bug hunting in many industrial problems. Unfortunately, testing is not a panacea. Indeed, since there is, in general, no way for a finite set of test cases to cover all possible scenarios, errors may remain undetected.

There are also methods that can ensure the full correctness of a system. Those methods, also called *formal methods*, use mathematical techniques to check whether the system will behave correctly for all possible scenarios. There are several mathematical representations for a system. In this thesis, we will consider (extensions of) *Transition Systems*. The behaviors of a transition system can be represented by (possibly infinite) sequences of state changes and time stamps, which we call *executions*. The relation between successive states being obtained by a so-called *transition relation*. This relation may not be finite; it may also be implicit.

There is a long history of formal methods, going from logical proofs and invariants to model checking [BK08]. In this thesis, we focus on the second approach. It consists in checking that each behavior of the system satisfies a given requirement by exploring its state-space. In early work on the subject, requirements are often expressed in some temporal logic such as Linear Temporal

Logic [Pnu77], or computational Tree Logic [CE81]. Those logics extend classical Boolean logics with (quantification of) temporal operators that allows us to reason on the temporal dimension of a given execution.

It can be shown that solving the model checking problem boils down to compute a (repeated) set of reachable states [CGP99]. A simple state-space exploration technique starts the exploration from the set of initial states and then adds new reachable states by applying the reachability relation. If the number of states is finite, repeating this operation will eventually produce a stable set, that is the set of reachable states of the system. However, even for simple systems, finite-state spaces can be much too large to be computed and represented with realistic amounts of computer resources. For several decades now, researchers have been looking at ways to reduce the computational burden associated with these state space exploration based techniques.

A first family of strategies developed for coping with large state spaces is to exploit similarities and repetitive information. Among such techniques, one finds the so-called partial reduction [WG93,FG05]. This approach avoids the exploration of sequences of states by showing that their effect is already captured by another sequence. Another technique is called bisimulation reduction [DPP04]. It exploits equivalence classes of bisimilar states (i.e., states that generate the same behaviors) to reduce the state space. Predicate abstraction techniques [BMR05] extend bisimulation reduction by abstracting sets with a given predicate that subsumes their behaviors. The difficulty being to find the predicate that do not blow up the set of behaviors artificially. Predicate abstraction based techniques can be combined with CounterExample approaches used to calibrate the precision of the abstraction [CV03].

In addition to compute state-space, one of the major difficulties in model checking is to represent sets of state in an efficient way. One of the very first family of strategies developed for coping with large state spaces is based on symbolic methods which use symbolic representation to manipulate set of states implicitly rather than explicitly. Symbolic methods have managed to broaden the applicability of simple analysis methods, such as state space exploration, to systems with impressively large sets of states. One of the most used symbolic representation is known as Binary Decision Diagrams (BDD in short) [Bry92]. In BDDs, the states of the system are encoded with fixed-length bit vectors. In such a context, a finite set of states can be viewed as the set of solutions of a Boolean formula for which a BDD provides a representation that is often more compact than conjunctive or disjunctive normal form. This representation, algorithmically easy to handle, allows to efficiently represent the regular structure that often appears in the set of reachable states of finite state-transition systems. The BDD-based approach has been used to verify systems with more than 10^{20} reachable states [BCM+92], and it is now well-admitted that Boolean formal verification of large-size systems can be performed. Over the last decade, BDD have been replaced (or combined with) logical representation. Those consists in representing the sequence of states via formulas, and then use a sat-solvers to check for a reachable state [BCCZ99,GPS02].

For two decades, logics and formal models did not exploit and model informations such as real-time or probabilities. This is however needed to reason large class of systems such as Embedded systems, Cyber physical systems, or systems biology. There, one is more interested in computing the level of energy needed to stay above a certain threshold, or the time needed to reach a given state. Motivated by this observation, the research community extended transitions systems with the ability to handle quantitative features. This includes, e.g., the formalism of timed automata [A.99] that exploits real-time informations to guide the executions, stochastic systems that can capture uncertainty in the executions, or weighted automata which permits to quantify the weight of a set of transitions [DG07]. In a similar fashion, LTL/CTL were extended with timed and quantitative informations. Those formalisms have been largely discussed in the literature, and have extended to other classes such as energy automata, or hybrid systems. It has been observed that reasoning on quantities amplifies the state-space explosion problem. However, tools such as UPPAAL or PRISM provided efficient approaches to partly overcome those problems. In this work, we focus on the stochastic aspects.

1.1 The stochastic world: towards SMC

Among the prominent extensions of transitions sytems, one finds quantitative sytems whose transitions are equipped with a probability distribution. This category includes, e.g., both discrete and continuous timed Markov Chains[1]. Our main interest will be in computing the probability to satisfy a given property of a stochastic system. This quantification replaces the Boolean world and permits us to quantify the impact of changes made on a given system.

Like classical transition systems, quantitative properties of stochastic systems are usually specified in linear temporal logics that allow one to compare the measure of executions satisfying certain temporal properties with thresholds. The model checking problem for stochastic systems with respect to such logics is typically solved by a numerical approach that, like state-space exploration, iteratively computes (or approximates) the exact measure of paths satisfying relevant subformulas. The algorithm for computing such measures depends on the class of stochastic systems being considered as well as the logics used for specifying the correctness properties. Model checking algorithms for a variety of contexts have been discovered [BHHK03,CY95,CG04] and there are mature tools (see e.g. [KNP04,CB06]) that have been used to analyze a variety of systems in practice.

Despite the great strides made by numerical model checking algorithms, there are many challenges. Numerical algorithms work only for special systems that have certain structural properties. Further the algorithms require a lot of time and space, and thus scaling to large systems is a challenge. In addition, the logics for which model checking algorithms exist are extensions of classical temporal

[1] As we shall see later, stochastic systems may deal with additional quantities such as real-time.

logics, which are often not the most popular among engineers. Finally, those numerical techniques do not allows us to consider extended stochastic models whose semantics also depends on other quantities such as real-time, or energy.

Another approach to verify quantitative properties of stochastic systems is to *simulate* the system for finitely many runs, and use techniques coming from the area of statistics to infer whether the samples provide a *statistical* evidence for the satisfaction or violation of the specification [YS02]. The crux of this approach is that since sample runs of a stochastic system are drawn according to the distribution defined by the system, they can be used to get estimates of the probability measure on executions. Those techniques are known under the name of Statistical Model Checking (SMC).

The SMC approach enjoys many advantages. First, these algorithms only require that the system be simulatable (or rather, sample executions be drawn according to the measure space defined by the system). Thus, it can be applied to larger class of systems than numerical model checking algorithms including black-box systems and infinite state systems. Second the approach can be generalized to a larger class of properties, including Fourier transform based logics. Finally, the algorithm is easily parallelizable, which can help scale to large systems. In case the problem is undecidable or too complex, SMC is often the only viable solution. SMC algorithms have been implemented in a series of tools such as Ymer [You05a], PRISM [KNP11], or UPPAAL [DLL+11]. Recently, we have implemented a series of SMC techniques in a flexible and modular toolset called Plasma Lab [BCLS13]. In the next section, we introduce the basic SMC algorithm and the major challenges that will be tackled at Isola.

2 Statistical model checking: a brief technical introduction

We consider a set of states S and a time domain $T \subseteq \mathbb{R}$. We first introduce the general definition of stochastic systems.

Definition 1 (Stochastic system). *A* stochastic system *over S and T is a family of random variables $\mathcal{X} = \{X_t \mid t \in T\}$, each random variable X_t having range S.*

The definition of a stochastic system as a family of random variables is quite general and includes systems with both continuous and discrete dynamics. In this thesis, we will focus our attention on a limited, but important, class of stochastic system: stochastic discrete event systems, which we note $\mathcal{S} = (S, T)$. This class includes any stochastic system that can be thought of as occupying a single state for a duration of time before an event causes an instantaneous state transition to occur. An *execution* for a stochastic system is any sequence of observations $\{x_t \in S \mid t \in T\}$ of the random variables $X_t \in \mathcal{X}$. It can be represented as a sequence $\omega = (s_0, t_0), (s_1, t_1), \ldots, (s_n, t_n) \ldots$, such that $s_i \in S$ and $t_i \in T$, with time stamps monotonically increasing, e.g. $t_i < t_{i+1}$. Let $0 \leq i \leq n$, we denote

$\omega^i = (s_i, t_i), \ldots, (s_n, t_n)$ the suffix of ω starting at position i. Let $\overline{s} \in S$, we denote $Path(\overline{s})$ the set of executions of $\mathcal{X}$ that starts in state $(\overline{s}, 0)$ (also called initial state) and $Path^n(\overline{s})$ the set of executions of length n.

In [You05a], Youness showed that the executions set of a stochastic system is a measurable space, which defines a probability measure μ over $Path(\overline{s})$. The precise definition of μ depends on the specific probability structure of the stochastic system being studied.

Requirements. In this thesis, except if explicitly mentioned,Properties over traces of $\mathcal{S}ys$ are defined via the so-called Bounded Linear Temporal Logic (BLTL). BLTL restricts Linear Temporal Logic by bounding the scope of the temporal operators. The syntax of BLTL is defined as follows:

$$\phi = \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi \mid \mathrm{F}^{\leq t}\phi \mid \mathrm{G}^{\leq t}\phi \mid \phi\,\mathrm{U}^{\leq t}\phi \mid \mathrm{X}\phi \mid \alpha$$

$\vee, \wedge$ and $\neg$ are the standard logical connectives and α is a Boolean constant or an atomic proposition constructed from numerical constants, state variables and relational operators. X is the *next* temporal operator: $\mathrm{X}\phi$ means that ϕ will be true on the next step. F, G and U are temporal operators bounded by time interval $[0, t]$, relative to the time interval of any enclosing formula. We refer to this as a *relative interval*. F is the *finally* or *eventually* operator: $\mathrm{F}^{\leq t}\phi$ means that ϕ will be true at least once in the relative interval $[0, t]$. G is the *globally* or *always* operator: $G^{\leq t}\phi$ means that ϕ will be true at all times in the relative interval $[0, t]$. U is the *until* operator: $\psi \mathrm{U}_{\leq t}\phi$ means that in the relative interval $[0, t]$, either ϕ is initially true or ψ will be true until ϕ is true. Combining these temporal operators creates complex properties with interleaved notions of *eventually* (F), *always* (G) and *one thing after another* (U).

Verifying BLTL properties: a simulation approach Consider a stochastic system (S, T) and a property ϕ. *Statistical model checking* refers to a series of simulation-based techniques that can be used to answer two questions: (1) **Qualitative:** Is the probability that (S, T) satisfies ϕ greater or equal to a certain threshold? and (2) **Quantitative:** What is the probability that (S, T) satisfies ϕ? Contrary to numerical approaches, the answer is given up to some correctness precision. As we shall see latter, SMC solves those problems with two different approaches, while classical numerical approaches only solve the second problem, which implies the first one, but is harder.

In the rest of the section, we overview several statistical model checking techniques. Let B_i be a discrete random variable with a Bernoulli distribution of parameter p. Such a variable can only take 2 values 0 and 1 with $Pr[B_i = 1] = p$ and $Pr[B_i = 0] = 1 - p$. In our context, each variable B_i is associated with one simulation of the system. The outcome for B_i, denoted b_i, is 1 if the simulation satisfies ϕ and 0 otherwise. The latter is decided with the help of a monitoring[2]

[2] This thesis is not concerned with the definition of efficient monitoring procedures.

procedure[HR02]. The objective of an SMC algorithm is to generate simulations and exploit the Bernouili outcomes to extract a global confidence on the system.

In the next subsections, we present three algorithms used in history work on SMC to solve both the quantitative and the qualitative problems. Extension of those algorithms to unbounded temporal operators [SVA05,HCZ11] and to nested probabilistic operators exist [You05b]. As shown in [JKO+07] those extensions or debatable and often slower than their . Consequently, we will not discuss them.

2.1 Qualitative answer using statistical model checking

The main approaches [You05a,SVA04] proposed to answer the qualitative question are based on *hypothesis testing*. Let $p = Pr(\phi)$, to determine whether $p \geq \theta$, we can test $H : p \geq \theta$ against $K : p < \theta$. A test-based solution does not guarantee a correct result but it is possible to bound the probability of making an error. The *strength* (α, β) of a test is determined by two parameters, α and β, such that the probability of accepting K (respectively, H) when H (respectively, K) holds, called a Type-I error (respectively, a Type-II error), is less or equal to α (respectively, β).

A test has *ideal performance* if the probability of the Type-I error (respectively, Type-II error) is exactly α (respectively, β). However, these requirements make it impossible to ensure a low probability for both types of errors simultaneously (see [You05a] for details). A solution to this problem is to relax the test by working with an *indifference region* (p_1, p_0) with $p_0 \geq p_1$ ($p_0 - p_1$ is the *size of the region*). In this context, we test the hypothesis $H_0 : p \geq p_0$ against $H_1 : p \leq p_1$ instead of H against K. If the value of p is between p_1 and p_0 (the indifference region), then we say that the probability is sufficiently close to θ so that we are indifferent with respect to which of the two hypotheses K or H is accepted. The thresholds p_0 and p_1 are generally defined in terms of the single threshold θ, e.g., $p_1 = \theta - \delta$ and $p_0 = \theta + \delta$. We now need to provide a test procedure that satisfies the requirements above. In the next two subsections, we recall two solutions proposed by Younes in [You05a,You06].

Single Sampling Plan. This algorithm is more for history than for direct usage. However, it is still exploited in subsequent algorithms. To test H_0 against H_1, we specify a constant c. If $\sum_{i=1}^{n} b_i$ is larger than c, then H_0 is accepted, else H_1 is accepted. The difficult part in this approach is to find values for the pair (n, c), called a *single sampling plan (SSP in short)*, such that the two error bounds α and β are respected. In practice, one tries to work with the smallest value of n possible so as to minimize the number of simulations performed. Clearly, this number has to be greater if α and β are smaller but also if the size of the indifference region is smaller. This results in an optimization problem, which generally does not have a closed-form solution except for a few special cases [You05a]. In his thesis [You05a], Younes proposes a binary search based algorithm that, given p_0, p_1, α, β, computes an approximation of the minimal value for c and n.

Sequential Probability Ratio Test (SPRT). The sample size for a single sampling plan is fixed in advance and independent of the observations that are made. However, taking those observations into account can increase the performance of the test. As an example, if we use a single plan (n, c) and the $m > c$ first simulations satisfy the property, then we could (depending on the error bounds) accept H_0 without observing the $n-m$ other simulations. To overcome this problem, one can use the *sequential probability ratio test (SPRT in short)* proposed by Wald [Wal45]. The approach is briefly described below.

In SPRT, one has to choose two values A and B ($A > B$) that ensure that the strength of the test is respected. Let m be the number of observations that have been made so far. The test is based on the following quotient:

$$\frac{p_{1m}}{p_{0m}} = \prod_{i=1}^{m} \frac{Pr(B_i = b_i | p = p_1)}{Pr(B_i = b_i | p = p_0)} = \frac{p_1^{d_m}(1-p_1)^{m-d_m}}{p_0^{d_m}(1-p_0)^{m-d_m}}, \tag{1}$$

where $d_m = \sum_{i=1}^{m} b_i$. The idea behind the test is to accept H_0 if $\frac{p_{1m}}{p_{0m}} \geq A$, and H_1 if $\frac{p_{1m}}{p_{0m}} \leq B$. The SPRT algorithm computes $\frac{p_{1m}}{p_{0m}}$ for successive values of m until either H_0 or H_1 is satisfied; the algorithm terminates with probability 1[Wal45]. This has the advantage of minimizing the number of simulations. In his thesis [You05a], Younes proposed a logarithmic based algorithm SPRT that given p_0, p_1, α and β implements the sequential ratio testing procedure.

SPRT has been largely used in the formal methods area. In this thesis, we shall show that the approach extends to a much larger class of problems that the one originally foreseen.

2.2 Quantitative answer using statistical model checking and Estimation

In the case of estimation, existing SMC algorithms rely on classical Monte Carlo estimation. More precisely, they calculate a priori the required number of simulations according to a Chernoff bound [Oka59] that allows the user to specify an error ε and a probability δ that the estimate $\hat{p}$ will not lie outside the true value $\pm\varepsilon$. Given that a system has true probability p of satisfying a property, the Chernoff bound ensures $\mathrm{P}(\mid \hat{p} - p \mid \geq \varepsilon) \leq \delta$. Parameter δ is related to the number of simulations N by $\delta = 2e^{-2N\varepsilon^2}$ [Oka59], giving

$$N = \left\lceil (\ln 2 - \ln \delta)/(2\varepsilon^2) \right\rceil. \tag{2}$$

2.3 On expected number of simulations

The efficiency of the above algorithms is characterized by the number of simulations needed to obtain an answer. This number may change from executions to executions and can only be estimated (see [You05a] for an explanation). However, some generalities are known. For the qualitative case, it is known that, except for some situations, SPRT is always faster than SSP. When $\theta = 1$ (resp. $\theta = 0$) SPRT degenerates to SSP; this is not problematic since SSP is known to

be optimal for such values. Monte Carlo can also be used to solve the qualitative problem, but it is always slower than SSP [You05a]. If θ is unknown, then a good strategy is to estimate it using Monte Carlo with a low confidence and then validate the result with SPRT and a strong confidence.

2.4 Challenges

Unfortunately, the SMC approach we introduced above is not a panacea and many important classes of systems and properties are still out of its scope. This includes, e.g., unbounded properties. Moreover, In addition, SMC still indirectly suffers from an explosion linked to the number of simulations needed to converge when estimating small probabilities, a.k.a rare events. Finally,the approach has not yet been lifted to a professional toolset directly usable by industry people. Consequently, it remains unclear whether the approach can handle applications that are beyond the academic world.

This session proposes solutions to those challenges.

3 Contribution to the track

This tracks contains several contributions to improve the weakness of SMC pointed in the previous section. Those are divided into three main categories, that are 1. imrpoving SMC algorithm in terms of speeds or models that can be handled, improving tooling, and applying SMC to new categories. A summary is given here after.

3.1 On extension of SMC algorithms

- Statistical model checking avoids the exponential growth of states associated with probabilistic model checking by estimating probabilities from multiple executions of a system and by giving results within confidence bounds. Rare properties are often important but pose a particular challenge for simulation-based approaches, hence a key objective for SMC is to reduce the number and length of simulations necessary to produce a result with a given level of confidence. In the literature, one finds two techniques to cope with rare events: *Importance Sampling* (IS) and *importance Splitting* (IP). One of the majors problems with IS simulation is that it does not yield 0/1-outcomes, as assumed by the existing hypothesis tests, but likelihood ratios that are typically close to zero, but may also take large values. In [RdBS], the authors consider two possible ways of combining IS and SMC. One involves an easily applicable IS-scheme that yields likelihood ratios with bounded support when applied to a certain (nontrivial) class of models. The other involves a particular hypothesis testing scheme that does not require a-priori knowledge about the samples, only that their variance is estimated well.

– One of the major limitations of SMC is that it is limited to bounded properties, i.e., properties that can be evaluated on finite traces. A series of recent work shows that this situation can be improved for several classes of systems/property. In [Kre], the author survey statistical verification techniques aiming at linear properties with unbounded or infinite horizon, as opposed to properties of runs of fixed length. Moreover, the author also discusses when it is possible to statistically estimate linear distances between Markov chains.
– One of the major difficulties of stochastic model checking is to obtain a model on which SMC can be applied. In [JLL+], the authors introduce feedback-control statistical system checking (FCSSC), a new approach to statistical model checking that exploits principles of feedback-control for the analysis of cyber-physical systems (CPS). FC-SSC uses stochastic system identification to learn a CPS model, importance sampling to estimate the CPS state, and importance splitting to control the CPS so that the probability that the CPS satisfies a given property can be efficiently inferred. They show the applicability of the approach on concrete applications.
– It is crucial for accurate model checking that the model be a complete and faithful representation of the system. Unfortunately, this is not always possible, mainly because of two reasons: (i) the model is still under development and (ii) the correctness of implementation of some modules is not established. In [AM], the author examinates circumstances, is it still possible to get correct answers for some model checking queries in the case of PCTL and Markov Chains.

3.2 On tools

– In [LST], the authors present an overview of Plasma Lab, a modular statistical model checking (SMC) platform that facilitates multiple SMC algorithms, multiple modelling and query languages and has multiple modes of use. Plasma Lab may be used as a stand-alone tool with a graphical development environment or invoked from the command line for high performance scripting applications. Plasma Lab is written in Java for maximum cross-platform compatibility, but it may interface with tools and libraries written in arbitrary programming languages. Plasma Lab's API also allows it to be incorporated as a library within other tools.
– Streaming applications for mobile platforms impose high demands on a system's throughput and energy consumption. Dynamic system-level techniques have been introduced, to reduce power consumption at the expense of performance. We consider DPM (Dynamic Power Management) and DVFS (Dynamic Voltage and Frequency Scaling). The complex programming task now includes mapping and scheduling every task onto a heterogeneous multiprocessor hardware platform. Moreover, DPM and DVFS parameters must be controlled, to meet all throughput constraints while minimizing the energy consumption. In [AvdP], the authors experiment with an alternative

approach, based on stochastic hybrid games. Their main contribution is to compare simulation-based tools applied to this problematic.

3.3 On new applications

- In [tBLVL], the authors examinate the problem of applying SMC to systems with variability. They mostly focus on product lines paradigm. They report on the suitability of statistical model checking for the analysis of quantitative properties of product line models by an extended treatment of earlier work by the authors. The type of analysis that can be performed includes the likelihood of specific product behaviour, the expected average cost of products (in terms of the attributes of the products' features) and the probability of features to be (un)installed at runtime. They illustrate the feasibility of their framework by applying it to a case study of a product line of bikes.
- Scheduling and control of Cyber-Physical Systems (CPS) are becoming increasingly complex, requiring the development of new techniques that can effectively lead to their advancement. This is also the case for failure detection and scheduling component replacements. In [LdPB], the authors propose a technique that not only relies on machine learning classification models in order to classify component failure cases vs. non-failure cases, but also on real-time updating of the maintenance policy of the sub-system in question. The technique is implemented in UPPAAL.
- In [RS], the authors present a framework, fault maintenance trees (FMTs), integrating maintenance into the industry-standard formalism of fault trees. By translating FMTs to priced timed automata and applying statistical model checking, we can obtain system dependability metrics such as system reliability and mean time to failure, as well as costs of maintenance and failures over time, for different maintenance policies. The approach is applied on two case studies from the railway industry: electrically insulated joints, and pneumatic compressors.
- Finally, the work in [Str] presents a panacea of applications for Statistical Model Checking on timed stochastic systems, while [Pel] make the link with genetic algorithms.

4 Conclusion

In this track, the authors have presented major advances for Statistical Model Checking. However, a lot remains to do. This include, e.g., better strategies to handle non determinism, more applications to real-life systems, or combining SMC with other approaches such as machine learning or testing.

References

[A.99] Rajeev A. Timed automata. In *CAV*, volume 1633 of *LNCS*. Springer, 1999.

[AM] Shiraj Arora and Panduranga Rao MV. Probabilistic model checking of incomplete models. In this proceedings.

[AvdP] Waheed Ahmad and Jaco van de Pol. Synthesizing energy-optimal controllers for multi-processor dataflow applications with uppaal. In this proceedings.

[BCCZ99] A. Biere, A. Cimatti, E. M. Clarke, and Y. Zhu. Symbolic model checking without bdds. In *TACAS*, volume 1579 of *LNCS*, pages 193–207. Springer, 1999.

[BCLS13] B. Boyer, K. Corre, A. Legay, and S. Sedwards. Plasma-lab: A flexible, distributable statistical model checking library. In *QEST*, volume 8054 of *LNCS*, pages 160–164, 2013.

[BCM+92] J. R. Burch, E. M. Clarke, K. L. McMillan, D. L. Dill, and L. J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Inf. and Comp.*, 98(2):142–170, 1992.

[BHHK03] Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Model-checking algorithms for continuous-time markov chains. *IEEE Trans. Software Eng.*, 29(6):524–541, 2003.

[BJK+05] M. Broy, B. Jonsson, J-P. Katoen, M. Leucker, and A. Pretschner, editors. *Model-Based Testing of Reactive Systems, Advanced Lectures The volume is the outcome of a research seminar that was held in Schloss Dagstuhl in January 2004*, volume 3472 of *Lecture Notes in Computer Science*. Springer, 2005.

[BK08] C. Baier and J-P. Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[BMR05] T. Ball, T. D. Millstein, and Sriram K. Rajamani. Polymorphic predicate abstraction. *ACM Trans. Program. Lang. Syst.*, 27(2), 2005.

[Bry92] R. Bryant. Symbolic boolean manipulation with ordered binary-decision diagrams. *ACM Computing Survey*, 24(3):293–318, 1992.

[CB06] F. Ciesinski and C. Baier. Liquor: A tool for qualitative and quantitative linear time analysis of reactive systems. In *Proc. of 3rd Int. Conference on the Quantitative Evaluation of Systems (QEST)*, pages 131–132. IEEE, 2006.

[CE81] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In *Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981.

[CG04] F. Ciesinski and M. Größer. On probabilistic computation tree logic. In *Validation of Stochastic Systems*, LNCS, 2925, pages 147–188. Springer, 2004.

[CGP99] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[CV03] E. M. Clarke and H. Veith. Counterexamples revisited: Principles, algorithms, applications. In *Verification: Theory and Practice, Essays Dedicated to Zohar Manna on the Occasion of His 64th Birthday*, volume 2772 of *LNCS*, pages 208–224. Springer, 2003.

[CY95] Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.

[DG07] Manfred Droste and Paul Gastin. Weighted automata and weighted logics. *Theor. Comput. Sci.*, 380(1-2):69–86, 2007.

[DLL+11] A. David, K.G. Larsen, A. Legay, Z.Wang, and M. Mikucionis. Time for real statistical model-checking: Statistical model-checking for real-time systems. In *CAV*, LNCS. Springer, 2011.

[DPP04] A. Dovier, C. Piazza, and A. Policriti. An efficient algorithm for computing bisimulation equivalence. *TCS*, 311(1-3), 2004.

[FG05] C. Flanagan and P. Godefroid. Dynamic partial-order reduction for model checking software. In *POPL*, pages 110–121. ACM, 2005.

[GPS02] G.Cabodi, P.Camurati, and S.Quer. Can bdds compete with sat solvers on bounded model checking? In *Proc. of 39th Design Automation Conference (DAC)*, pages 117–122. ACM, 2002.

[HCZ11] Younes H.., E. M. Clarke, and P. Zuliani. Statistical verification of probabilistic properties with unbounded until. In *Formal Methods: Foundations and Applications*, volume 6527 of *LNCS*, pages 144–160. Springer, 2011.

[HR02] K. Havelund and G. Rosu. Synthesizing monitors for safety properties. In *TACAS*, volume 2280 of *LNCS*, pages 342–356. 2002.

[JKO+07] D. N. Jansen, J-P. Katoen, M. Oldenkamp, M. Stoelinga, and I. S. Zapreev. How fast and fat is your probabilistic model checker? an experimental performance comparison. In *HVC*, volume 4899 of *LNCS*. Springer, 2007.

[JLL+] Cyrille Jegourel, Anna Lukina, Axel Legay, Scott Smolka, Radu Grosu, and Ezio Bartocci. Feedback control for statistical model checking of cyber-physical systems. In this proceedings.

[KNP04] M. Z. Kwiatkowska, G. Norman, and D. Parker. Prism 2.0: A tool for probabilistic model checking. In *QEST*, pages 322–323. IEEE, 2004.

[KNP11] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV'11*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[Kre] Jan Kretinsky. Survey of statistical verification of linear unbounded properties: Model checking and distances. In this proceedings.

[LdPB] Alexis LINARD and Marcos Luiz de Paula Bueno. Towards adaptive scheduling of maintenance for cyber-physical systems. In this proceedings.

[LST] Axel Legay, Sean Sedwards, and Louis-Marie Traonouez. Plasma lab: A modular statistical model checking platform. In this proceedings.

[Oka59] Masashi Okamoto. Some inequalities relating to the partial sum of binomial probabilities. *Annals of the Institute of Statistical Mathematics*, 10:29–35, 1959.

[Pel] Doron Peled. Automatic synthesis of code using genetic programming.

[Pnu77] A. Pnueli. The temporal logic of programs. In *Proc. 18th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 46–57, 1977.

[RdBS] Danil Reijsbergen, Pieter-Tjerk de Boer, and Werner Scheinhardt*. Hypothesis testing for rare-event simulation: limitations and possibilities. In this proceedings.

[RS] Enno Ruijters and Marielle Stoelinga. Better railway engineering through statistical model checking. In this proceedings.

[Str] Josef Strnadel. On creation and analysis of reliability models by means of stochastic timed automata and statistical model checking: Principle. In this proceedings.

[SVA04] K. Sen, M. Viswanathan, and G. Agha. Statistical model checking of black-box probabilistic systems. In *CAV*, LNCS 3114. Springer, 2004.

[SVA05] K. Sen, M. Viswanathan, and G. Agha. On statistical model checking of stochastic systems. In *CAV*, pages 266–280, 2005.

[tBLVL] Maurice ter Beek, Axel Legay, Andrea Vandin, and Alberto Lluch Lafuente. Statistical model checking for product lines. In this proceedings.

[Wal45] A. Wald. Sequential tests of statistical hypotheses. *Annals of Mathematical Statistics*, 16(2):117–186, 1945.

[WG93] P. Wolper and P. Godefroid. Partial-order methods for temporal verification. In *CONCUR*, volume 715 of *LNCS*. Springer, 1993.

[You05a] H. L. S. Younes. *Verification and Planning for Stochastic Processes with Asynchronous Events*. PhD thesis, Carnegie Mellon, 2005.

[You05b] H. L. S. Younes. Verification and planning for stochastic processes with asynchronous events. PhD thesis, Carnegie Mellon University, 2005.

[You06] H. L. S. Younes. Error control for probabilistic model checking. In *Proc. of 7th Int. Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI*, LNCS 3855, pages 142–156. springer-verlag, 2006.

[YS02] Håkan L. S. Younes and Reid G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *CAV*, LNCS 2404, pages 223–235. Springer, 2002.